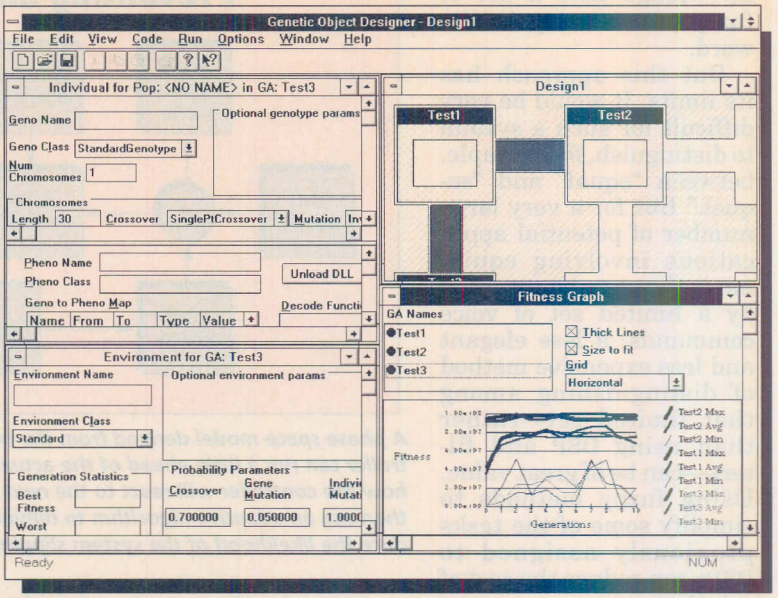
New software techniques boost the IQs of embedded systems

Fuzzy logic offers new approaches to old problems, using less software and fewer hardware resources. But this may mean giving up a general solution to implement a specific application.

Tom Williams, Senior Editor



Thomas Mann once remarked that there's something mysterious about names, "as if they give us the power of conjurers." Once we've named something it appears to take on the qualities suggested by the name, all reality to the contrary. For example, fuzzy logic appears to be imprecise, and neural networks suggest systems that function like brains. Genetic algorithms suggest artificial life and artificial intelligence and expert systems paint images of gurus dispensing wisdom. Despite what may be unreasonable expectations raised by such nifty sounding names, systems using techniques often grouped under the concept of artificial intelligence or expert systems can improve the "IQ" of computer-based systems—that is, help them perform their programs more efficiently with fewer resources.

When the computational power of algorithms running on fast silicon is mapped to bits of knowledge gleaned from human experience, we tend to call the systems intelligent. It's a backhanded way of complimenting ourselves. Still, this ability to combine computation with knowledge in an automatic system is cracking problems that once defied solutions.

Better use of resources

Fuzzy logic offers some new approaches to old problems that let them be solved with less software overhead and fewer hardware resources. The caveat is that these techniques may require you to give up a general solution to a problem in favor of implementing a specific application. One example demonstrated by Intel (Santa Clara, CA) in conjunction with fuzzy tool maker Inform (Aachen, Germany) is a speech-driven calculator. The calculator is trained to recognize a set of 16 words: the ten digits, four arithmetic operators, and the words equal and clear. The entire application fits in about 2 kbytes of code and runs on the 16-bit Intel 8XC196 family of microcontrollers.

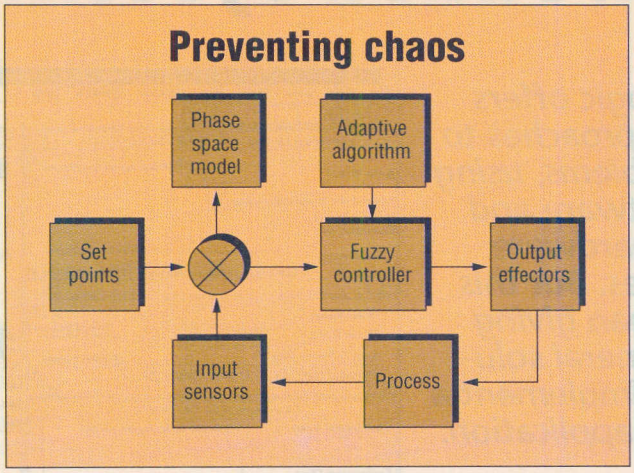
A major goal in the speech-recognition part of the application was to avoid using fast Fourier transforms which parse an utterance into a spectrum of frequency ranges. "We didn't want the conventional way of doing it with DSPs, FFTs and filters, and use a simple microcontroller," says Intel research engineer Mohammed Fennec.

The approach was to sample an utterance at about 10 kHz, obtaining up to 2048 samples stored in SRAM. Within the sampling rate, the utterance would

While the details of how to implement genetic algorithms in code are fairly esoteric, the availability of class libraries such as EOS from Man Machine Interfaces let programmers build them into applications. Windows based shells such as MMI's Genetic Object Designer (above) let people who understand their use but not the details of programming them employ the power of genetic algorithms in widely different disciplines.

show up as a simple graph of amplitude with higher density in regions of higher frequency. By dividing the graph into 24 squares, it's possible to characterize each square in terms of the density of samples in it. By inspecting the graph and deciding which regions carried the identifying characteristics for the word "equal," for example, the designer could write his knowledge into a set of fuzzy rules to recognize the word.

But this approach has its limits. It would be very difficult for such a system to distinguish, for example, between "equal" and "sequel." But for a very large number of potential applications involving equipment that can be actuated by a limited set of voice commands, a less elegant and less expensive method of distinguishing among that limited set—rather than using DSP and filters—can be of great value. Using fuzzy methods to simplify some of the tasks previously assigned to DSPs can reduce the cost of solutions.



A phase space model derived from the behavior of a fuzzy controller can run a little ahead of the actual controller and predict how the controller will react to the next set of inputs. It can then use an adaptive algorithm to diminish the error and reduce the likelihood of the system slipping into chaotic behavior.

Fuzzy logic for clear resolutions

The savings might lie in using a fixed-point DSP chip rather than a floating-point chip for a given application, or it might let you move an application to a less expensive processor or to use a current processor for additional tasks. According to Berthold Hellenthal, senior engineer with Inform, "We're trying to enhance the uses of the 196 (microcontroller) family by finding ways of using fuzzy logic to replace users' DSP applications."

One of these applications is in hard disk drive controllers where there's a premium on low cost, small size and high speed. Inform is looking at a single-chip solution which suggests using a microcontroller which already has on-chip peripherals for such things as high-speed I/O. If a fuzzy logic solution gives the required performance then a 16-bit MCU can offer a single-chip solution. "Fuzzy logic is in most cases so fast," says Hellenthal, "that what eats up most of the resources is peripheral activity or preprocessing and filtering in DSP applications. There are applications such as voice compression where the rest of the algorithms—the non-fuzzy part—are so complex that you do require a DSP."

But what if you have a microcontroller that has some DSP hardware capability? The 16-bit Motorola 68HC16Z1 microcontroller includes an on-chip A-D converter and instruction set support for low-frequency digital signal processing. A multiply and accumulate (MAC) unit can perform repetitive MAC operations and supports modulo addressing for setting up circular DSP buffers.

With the 68HC16Z1, Chris Hale, advanced microcontroller marketing engineer for Motorola, has used the MCU to digitize voice commands, filter them through a self-configuring low-pass filter and encode them into a series of frequency patterns that can be recognized using fuzzy inference rules. "When surrounded by noise, it becomes difficult to understand conversations," says Hale, "The problem becomes easy to solve if you know what will be said before the conversation starts."

Separating signal from noise

As with the speech-driven calculator, the vocabulary to be recognized is limited to a relatively small set of commands needed to operate a piece of equipment. The problem then becomes one of separating the signal from the noise and highlighting its unique characteristic to compare it with the set of expected utterances. "Without this prior knowledge, it becomes difficult to detect or understand," says Hale.

Because the same spoken words vary due to pitch, volume and accent, a recognition scheme must be flexible and capable of approximate reasoning. The incoming voice signal is converted from a time domain to a frequency domain using a fast Fourier transform. The fuzzy code determines the upper and lower edges of a desired frequency band and uses that information to determine the transfer function for the low-pass filter. To compensate for variations in pitch and volume the signal is normalized in terms of frequency and amplitude using fuzzy code. Finally, the normalized signal spectrum is compared to template spectra stored in ROM. A set of fuzzy rules determines the degree of match between an input signal and its corresponding template.

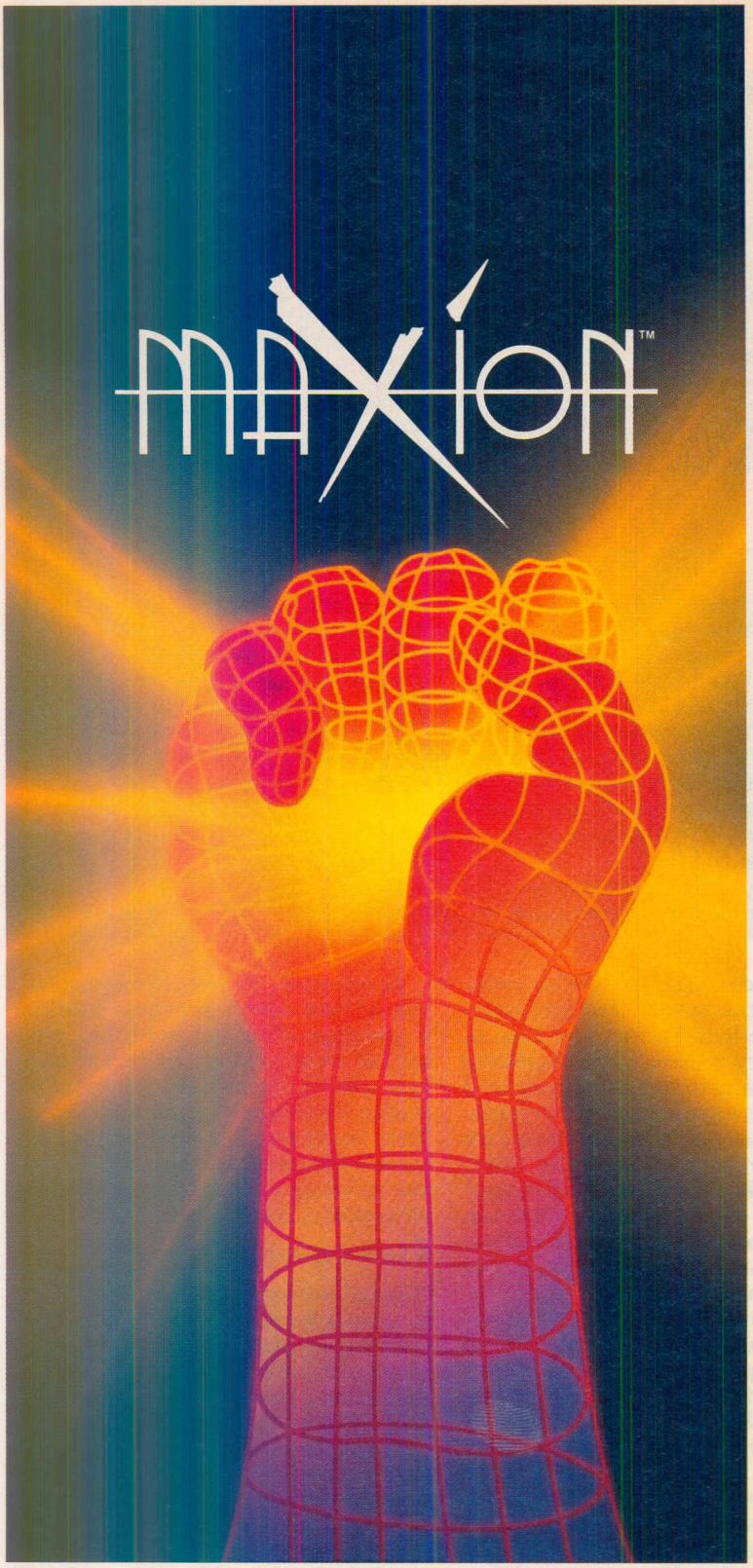
Order out of chaos

Control of non-linear systems and processes has always been a dicey business. The general strategy has been to derive an equation describing system behavior, identify linear portions of the equation and devise control strategies for them. This becomes increasingly difficult as the complexity of such systems increases. And the more complex and non-linear a system becomes, the greater the risk becomes of it exhibiting chaotic behavior. "Chaos theory says that although not every complex system shows chaotic behavior, the more complex a system is, the more likely it will show chaotic regions of behavior," says Ted Heske, senior engineer with the NCR Division of AT&T (Duluth, GA).

"The tendency of non-linear systems to give rise to chaotic, i.e. unpredictable, behavior," says Heske, "can possibly be countered by the ability of fuzzy systems to describe types of non-linear behavior without second- and third-order differential equations. "Where you have a chaotic time series, it's possible to derive non-linear fuzzy 'equations' and if you then iterate those equations you have some degree of predictability."

The fuzzy "equations" here are not equations in the classic sense, but sets of rules and membership

We POWER Success!



- We **POWER** the world's most advanced missile testing system.
- We **POWER** the world's most sophisticated attack helicopter simulator.
- We **POWER** the world's most advanced weather radar system...NEXRAD.
- We **POWER** the world's fastest growing financial trading information system.
- We **POWER** the world's most efficient subway system...Hong Kong.

WE ARE CONCURRENT COMPUTER CORPORATION.

Real Time and Beyond...multimedia, interactive video, medical imaging and mission critical on-line transaction processing ...all with our MAXION™ multiprocessor system — systems that deliver all the performance and power you pay for and demand in an open system environment.

Experience MAXION multiprocessing power for yourself. For a free CD-ROM demo, call Concurrent at 1-800-631-2154 or write to us at Concurrent Computer Corporation, Dept. MAX, 2 Crescent Place, Oceanport, NJ 07757.

CONCURRENT COMPUTER CORPORATION™

POWER. MAXION™ POWER!

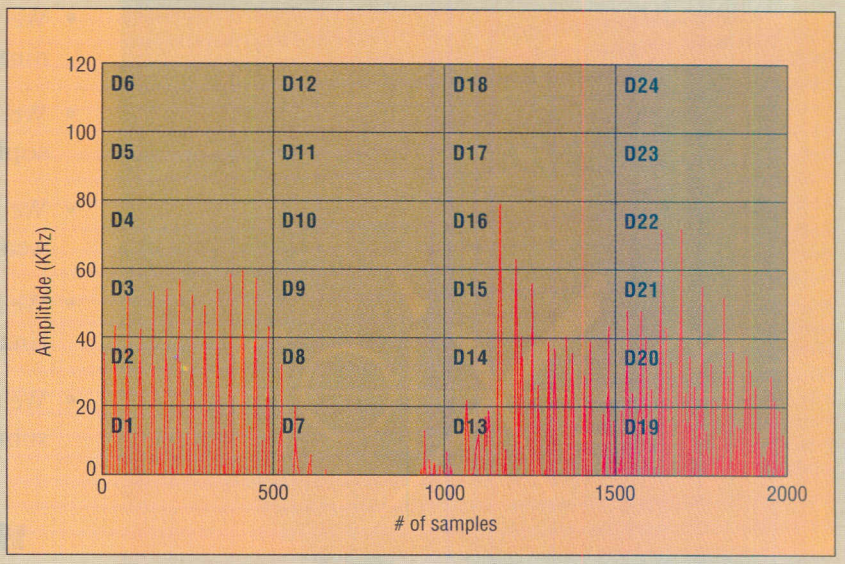
Circle 44

Recognizing words. . . artificially speaking

To recognize a small set of words for a calculator, Intel was able to divide the plot of a voice command into regions that could be characterized by density (low, medium and high). Four squares along the X axis show frequency over portions of the sampling time and six squares on the Y axis characterize loudness at various times during the utterance.

It was then possible to assign membership functions reflecting the density of each square and to write rules characterizing each word. Four "takes" of 2048 samples are used to obtain a characteristic signature of a word. But—and this is the point where human knowledge enters the recognition process—the machine doesn't establish the recognition criteria automatically. "We're not looking at the sine waves as the data holders," says Fennec, "just looking at the squares."

It's up to a person to examine the graph of the sampled utterance, decide what constitutes the identifying characteristics and write rules that uniquely describe them. By inspecting multiple plots of an utterance and deciding which regions carried the identifying characteristics for the word "equal," for example, the designer can write his knowledge into a set of fuzzy rules to recognize the word.



While all rules in the rule base combine to form the final decision, a typical rule for "equal" is: IF D8 is Low AND D9 is VeryLow AND D10 is VeryLow AND D13 is Low AND D19 is Low AND D20 is Medium AND D21 is Medium AND D21 is Meduim AND D7 is VeryHigh; THEN Probability of EQUAL is VeryHigh AND Probability of Four is Low and Probability of AllOther is VeryLow.

The fuzzy inference would give a crisp output value for a given word. In cases where there are two competing words, the most probable word, i.e. the one with the highest value (90 percent or better) is selected. "It's kind of a combination of fuzzy logic and probability," says Fennec, "To write the rules, a person must get an intuitive feeling, your own knowledge base. And you don't have that knowledge until you examine the data with your own eyes."

functions describing the behavior of a system. The same set of input conditions will, in all cases, yield the same outputs; so the behavior model is predictable.

"You can produce a fuzzy model in much the same way the neural net guys do using model-free estimation techniques," says Heske, "by running input 'training' data into the primary control system and correlating its outputs with the inputs to build a fuzzy model. The fuzzy model would exist alongside the primary control strategy, outputs of which are used to derive the model. Then you could use the fuzzy model to predict the next elements in the series.

By stepping the fuzzy model ahead of the actual control systems, it's possible to see how the control system will react to the next set of inputs. If that reaction displays behavior that's beyond an acceptable deviation from what's expected, feedback from the fuzzy model can be used to curb the chaotic behavior. Inputs to the system are presented first to the fuzzy model and then to the primary control. If the inputs to the model and to the primary control system were presented on a graph, they'd appear as a phase space plot (two curves separated by some phase angle). A difference in output from the model that varied too widely from the expected output would be fed back into the system to dampen chaotic behavior.

Heske says his group is investigating realtime prediction, but at present, pre-derived models are used. The term realtime is of course relative. "The key question is what happens when your control loop becomes too long, if you violate what control engineers call the 'phase margin/gain margin product' In some cases you can produce a chaotic time series. If you can null that out with an intelligent algorithm, you've achieved smarter control."

In this case, though, the smarts of the system aren't previous human knowledge of its behavior, but empirical data of that behavior cast into a model that's used to predict it under real-world circumstances. That could possibly be described as knowledge generated and used by the machine. It might even qualify as artificial intelligence.

Evolving optimal solutions

Neural networks are basically tools to search for structure in data. In a model-free mode, they simply digest large gobs of data and reveal some underlying patterns. In error backpropagation they're trained to discern how incoming data fits into patterns that have already been presented to the system so that the new data can be classified.

The ability to establish associative relationships between patterns at the numerical level gives them what University of West Florida pro-

8ns BiCMOS 1Megs from Motorola. Everything else is dead in the water.

Motorola 0.5 micron BiCMOS SRAMs demonstrate a simple evolutionary principle: survival of the fastest.

With 8ns access times at 1Meg densities and 12ns at 4M, nothing else even comes close enough to compare – for speed *and* density.

MCM101524	MCM101525	MCM6726A	MCM6728A
1M x 4 bit	2M x 2 bit	128K x 8 bit	256K x 4 bit
12, 15ns	12, 15ns	8, 10, 12, 15ns	8, 10, 12, 15ns
MCM6729A*	**MCM6706A**	**MCM6705A**	**MCM6708A**
256K x 4 bit	32K x 8 bit	32K x 9 bit	64K x 4 bit
8, 10, 12, 15ns	8, 10, 12ns	10, 12ns	8, 10, 12ns
MCM6709A*	**MCM6706R**	**MCM6709R**	
64K x 4 bit	32K x 8 bit	64K x 4 bit	
8, 10, 12ns	6, 7, 8ns	6, 7, 8ns	

*Output Enable

Looking for even more speed? How about 6ns? That's the access time on our 256K BiCMOS Fast SRAMs.

Choose whichever speed-and-density combination is right for you. Either way, you'll get the built-in quality and reliability of Motorola's high volume, 0.5 micron manufacturing.

Reel in the power of our BiCMOS Fast SRAMs for your next design, and get ready to throw everything else back in the water.

To request technical information or a sample device, just mail in the coupon or FAX it to Motorola's Fast SRAM FAX line at **1-800-347-MOTO (6686)**.

CD9/94

Let's make some waves, Motorola.
Send me more on BiCMOS Fast SRAMs today.

Motorola, Inc. Fast SRAM Division, P.O. Box 1466, Austin, Texas 78767

Name _____ Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone _____

Application _____

Production Start Date _____ Estimated Usage: 1994 _____ 1995 _____

Configuration:

4 Meg ECL I/O
- ☐ 2 meg x 2 ☐ 12ns
- ☐ 1 meg x 4 ☐ 15ns

1 Meg TTL I/O
- ☐ 256K x 4 ☐ 8ns
- ☐ 128K x 8 ☐ 10ns
- ☐ 12ns

256K TTL I/O
- ☐ 64K x 4 ☐ 6ns ☐ 10ns
- ☐ 32K x 8 ☐ 7ns ☐ 12ns
- ☐ 32K x 9* ☐ 8ns

*10 & 12ns Only

If you like what's new, wait 'til you see what's next.

MOTOROLA

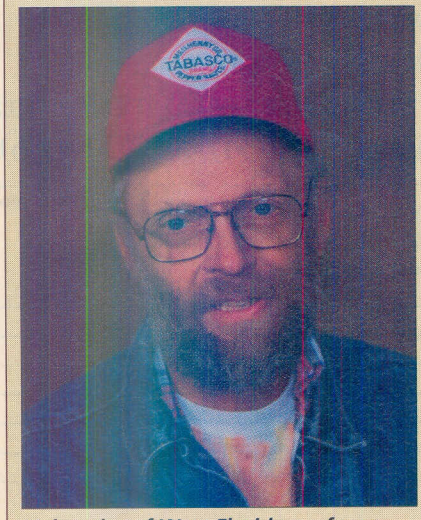
Computers aren't really intelligent

Are neural networks implemented in computers analogs of the human nervous system? Can they be considered intelligent?

Professor James Bezdek, whose career has been devoted to neural network, fuzzy logic and intelligent systems research, would caution us against what he calls the seductive semantics of such phrases as self-organizing, adaptive, genetic and learning. What are we doing when we design systems that incorporate non-numeric knowledge "tidbits" into computational models? While not denying that such systems have very definite uses for problem-solving, Bezdek proposes a kind of taxonomy of definitions.

The scheme consists of nine nodes starting with computational neural networks which can exhibit what he calls computational intelligence and ranges through artificial up to human biological intelligence.

"A computational neural network (CNN)," he says, "is any computational model that takes its inspiration from biology, and used for computational pattern recognition (CPR)." CPR



University of West Florida professor Jim Bezdek warns about being seduced by biological terminology into thinking that what automata do is real intelligence.

is one of many alternatives for numerical pattern recognition. Others include fuzzy and statistical pattern recognition models. CNNs operate solely on numerical data such as that obtained from sensors and computational pattern recognition (CPR) produces numerical results that reveal structure in sensor data.

"Computational intelligence (CI) begins to emerge," says Bezdek, "when the system exhibits computational adaptivity and fault tolerance, and when its speed and error rates approximate those of human performance."

The mid-level of Bezdek's taxonomy, the artificial level, requires the addition of knowledge tidbits to the computational model. "The CNN becomes an artificial neural network (ANN) when (and only when) enough knowledge tidbits are added to the CNN to endow it with rudimentary associative memory." But knowledge tidbits aren't what we really regard as knowledge.

To illustrate the difference, Bezdek hands a banana to a blindfolded man and asks him to identify it. Then he asks what kind of spider he's thinking of (tarantula), what song ("Hey, Mister Tallyman") and what signer (Bellafonte). "That's an example of how biological knowledge flows automatically from one thing to another," he says.

"An artificially intelligent system would be like a bunch of neural networks—one trained about fruit, and another about different kinds of spiders—somehow linked to one another (as they are in the brain). If you take the bottom row and add knowledge tidbits, you get what I regard as artificial neural networks, artificial pattern recognition—like syntactic pattern recognition—and artificial intelligence. The concept of knowledge tidbits just emphasizes the real differences between computational schemes that try to use little pieces of information and real knowledge."

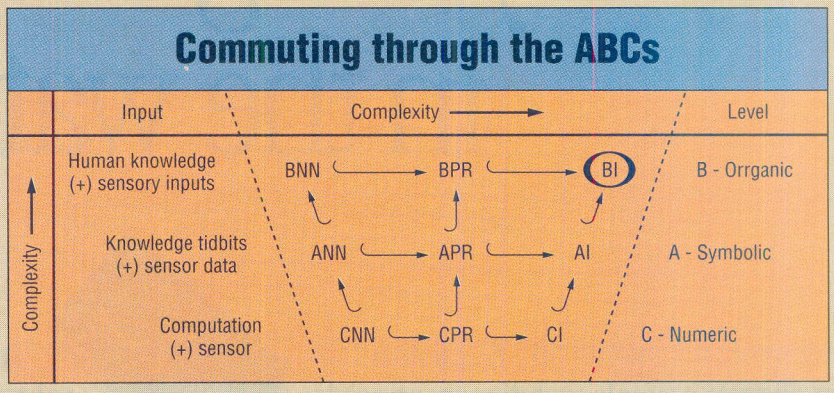
Ideal matchmaker

"Fuzzy models seem particularly well suited for combining knowledge tidbits with computational models," Bezdek says, "because they can accommodate both numerical and semantic information."

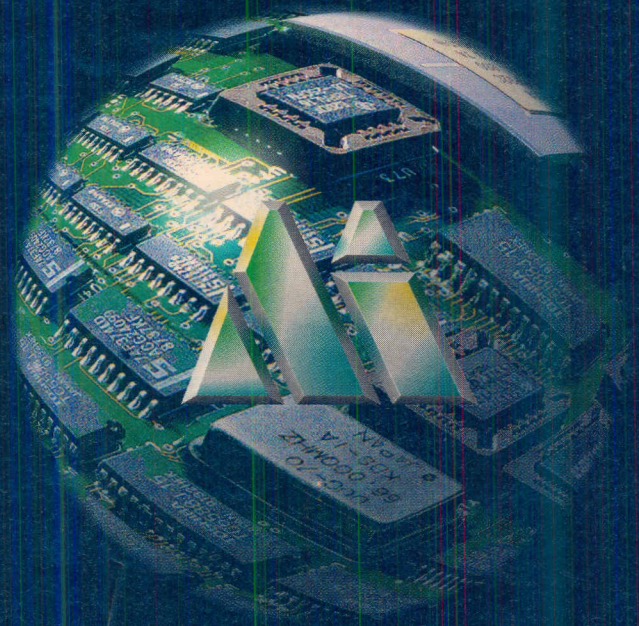
A fuzzy model subsumes a great deal of data within its membership functions which are given linguistic names. This grouping of data under linguistic names lets them be manipulated using a simple grammar and yet commands a very large amount of combinatorial complexity.

Still, the actual amount of knowledge contained in such a model remains relatively small. The expertise in a fuzzy system is fixed in the rule base which is similar to algorithms. "An algorithm is just an algorithm," says Bezdek, "it can't learn. This idea that neural networks learn is just stupid. They don't learn. If neural networks learn then Newton's method learns.

"Our assumption is that each neuron (in the brain) does some numerical computing—this gives rise to the hope that computers can be used to imitate this structure," he says.



The ABCs of neural nets. Computational (C) neural nets (CNNs) are the basic building blocks for artificial intelligence (AI) systems and are compared with biological (B) type systems which combine sensory inputs with human knowledge. The non-numeric content of AI systems, according to Bezdek's definition consists of "knowledge tidbits," small pieces of symbolic information that are associated with the computational capabilities of the neural net. In each case, pattern recognition (PR) is a subset of an intelligent system.



A WORLD OF RESOURCES

WE'RE MICRO INDUSTRIES.

We provide electronics solutions that help you bring ideas to market...faster, more cost-effectively, and with world class quality.

In a world that values *"Achieving Tomorrow's Ideas Today"*, Micro Industries can provide the competitive edge. You supply the idea—the product concept—and we supply the electronics solution. Micro-processor based board-level solutions. In your choice of standard, semi-custom, or fully custom boards. In low, medium, or high volumes. And always, as the needs of your application dictate.

Converting your ideas to reality takes more than the right board in the right slot. It takes a careful assessment of the application's requirements, plus the full range of resources—design, manufacturing, and quality assurance—to deliver a product that meets and exceeds your customers' expectations.

At Micro Industries, we have a world of resources at your fingertips. A team of engineers who are the best in their field, fully conversant with the latest board and manufacturing technologies. Advanced manufacturing techniques with an automated surface mount and conventional production facility registered by QMI to ISO-9001. A full-time quality assurance staff dedicated to manufacturing products to their specifications and committed to maintaining designed in reliability by controlling all phases of the manufacturing process. And after the product is designed, we'll support it for its entire product cycle.

In addition, we manufacture our products in a facility that does not pollute the environment.

Our expertise comes from 15 years of delivering solutions to automotive, consumer, industrial controls, telecommunications and medical electronics companies. We've developed long-term, mutually beneficial strategic partnerships with these companies that help them compete effectively in their world markets.

We can do the same for you. And, you should know, a partnership with Micro Industries doesn't end at product shipment. We're with you for the long run, with on-going product support as your products evolve and change to meet emerging market requirements.

Just call Julie today, for more information on how we can help you compete in today's global market.

MICRO INDUSTRIES ®

8399 Green Meadows Dr. N. Westerville, Ohio 43081-9486
TEL: (800) 642-7603 FAX: (614) 548-6184

QMI REGISTERED • QUALITY • SYSTEM
REGISTERED TO ISO 9001-94

Circle 47

fessor Jim Bezdek has termed "computational intelligence." How distinct a structure you wish to extract from a mass of data, or how small a margin of error for classifying it you wish to achieve, is largely up to you. If you want higher precision you must supply more data and cycle it through the neural net more times. And you can never achieve 100 percent precision or zero error.

Genetic algorithms are analogous to neural nets in that they search for some optimal solution that will not be 100 percent. The difference is that they're looking for an optimal solution to some complex equation or algorithm with many parameters that can result in many possible solutions. Often, the effects of some parameter can be subtle in that a small change to one value can have profound effects on the result. (The cliché in chaos theory about the butterfly causing a hurricane halfway around the world comes to mind.) But beyond theory, as systems become more complex, the search for optimal or acceptable solutions will increasingly be directed to automated methods.

Biological roots

Genetic algorithms take their terminology from biological concepts, but run the risk of again luring us into what Jim Bezdek calls seductive semantics. Terms like fitness (as in survival of), population, gene, and chromosome evoke life-like qualities that may be misleading but at the same time descriptive of what's going on. If, for example, the universe of possible solutions to a problem is plotted in X-Y space, the curve is called the fitness function. That is, the optimum solution is the point on the X axis where Y is maximum. Finding that point or a point that's "close enough" without exhaustively solving for every possible combination of variables is what genetic algorithms do. You specify how "fit" you want the solution to be. The more fit you want it, the more processing is involved.

In general terms, a genetic algorithm begins by generating a number of solutions at random and plotting their fitness functions. This constitutes the initial population. Each point is known as a chromosome and the parameters that generated it are genes. The next step is to pick a pair of points—ones with high fitness values—and use them for the next stage of the search. In mutation, you select a point and perturb some of its components. In reproduction by crossover, you select two points called parents and randomly combine their components to create a new point.

Mutation is usually a way of searching locally, but Salvatore Mangano, president of Man Machine Interfaces (MMI—Melville, NY) stresses that "you have to look at the whole space," as well as the area right around the parent points, "because a small change can make a big difference." The search can continue by either breeding successive generations of solutions and evaluating their fitness, or by starting a whole new population and breeding it until a predefined optimal value is reached.

Genetics at work

Tools for applying genetic algorithms are now on the market. NeuralWare (Pittsburgh, PA) offers an add-on package to its neural network software that lets you define genetic algorithm-based learning strategies for neural nets. The package comes with tips on how to tune parameters and use evaluation functions. MMI offers two development tools aimed at expanding the use and understanding of genetic algorithms for finding optimum solutions to complex problems.

MMI's Evolutionary Object System (EOS) is a library of over 80 C++ classes that support application development using genetic algorithms. Libraries support complex combinatorial problems such as the traveling salesman problem and simulated parallel execution which lets several different genetic algorithms work on the same or varying problem domains. EOS is supplied in standard and professional versions for Borland C++ and Microsoft Visual C++. The professional versions supply full source code in addition to full documentation of each function along with introductory material on the theory of genetic algorithms.

Another product from MMI is the Genetic Object Designer, a point-and-click tool that runs under Microsoft Windows and lets you explore genetic algorithms with little or no programming. The user interface lets you set up the problem as a piece of C++ or to bring it in via a Windows dynamic link library (DLL). You can then set the parameter of the genetic algorithm to determine such things as population or individual mutation.

"You may want to keep some parameters narrow because you already know part of what you want," says Mangano. Dialog boxes let you view values showing, for example, best, worst and average fitness, or determine genetic operators. Users of EOS can move their classes into the Genetic Object Designer environment, but EOS isn't required.

There's also a debug facility that lets you set breakpoints on genetic algorithm events such as breaking whenever a new generation is created, or some population size is reached. Genetic algorithms, like neural networks, aren't always suited to realtime applications, but they're of definite utility in developing realtime and embedded code by virtue of their ability to simplify, classify and optimize solutions that can then be encoded for use in embedded systems.

From a practical point of view the only really interesting question is, "Does it work? And if so, how well?" The question of whether we've got an expert system here or just another dumb controller can be debated endlessly. After all, it takes knowledge to derive an abstract mathematical model. But what we call expert or knowledge-based systems contain bits of knowledge as humans see and feel it in linguistic and graphical terms. As far as embedded systems go, new techniques for incorporating these tidbits can now be used to make certain operations more efficient and flexible, and that's a plus in terms of cost and performance. They may or may not be intelligent, but they do appear to be a good idea. ■

For more information about the technologies, products or companies mentioned in this article, call or circle the appropriate number on the Reader Inquiry Card.

AT&T
Duluth, GA
(404) 623-7000 . . . . . Circle 225

Inform
Aachen, Germany
(49) 2408-94560 . . . . . Circle 226

Intel
Santa Clara, CA
(408) 765-8080 . . . . . Circle 227

Man Machine Interfaces
Melville, NY
(516) 249-4700 . . . . . Circle 228

Motorola
Austin, TX
(512) 891-3465 . . . . . Circle 229

NeuralWare
Pittsburg, PA
(412) 787-8222 . . . . . Circle 230